À propos des réveils parasites (« spurious wakeups »)

Illustration de l'exercice 3 mais sur producteur/consommateur: cas de 2 producteurs (P1, P2) et 1 consommateur C et N=1.

N.B. On omet producer_done pour raison de simplification

Le problème peut survenir avec les 7 étapes suivantes:

1) On débute avec P1:

SC slot_lock :

On a nslots > 0 donc pas d'attente

nslots passe de 1 à 0

put_item(i) qui dépose et met la file à jour

SC item_lock :

nitems = 1

signal(&items)

2) Puis ensuite P2 s'exécute:

SC slot_lock :

On a nslots == 0 donc P2 va dans la file d'attente de slot_lock

Et on libère la SC slot_lock

3) Puis ensuite C s'exécute:

SC item_lock :

On a nitems > 0 donc pas d'attente

nitems passe de 0 à 1

get_item(i) lit et met la file à jour

SC slot_lock :

nslots passe de 0 è 1

signal(&slots)

4) Pendant que C est dans SC slot_lock, P1 revient en exécution :

SC slot_lock :

Mis dans file d'attente du mutex slot_lock ...

5) On revient à C, qui complète

SC slot_lock :

nslots passe de 0 à 1

signal(&slots)

6) On revient à P1 :

SC slot_lock :

On a nslots > 0 donc pas d'attente

nslots passe de 1 à 0

put_item(i) qui dépose et met la file à jour

SC item_lock :

nitems = 1

signal(&items)

7) Finalement, P2 revient suite au signal de C :

Puisque on a un if et non un while, P2 s'en va à la ligne 67 au lieu de revenir faire le test comme ça aurait été le cas avec un while (nslots <= 0).

Donc P2 va produire dans un tampon plein.

N.B. Si P2 était passer avant, on aurait pas eu le problème, mais on ne peut savoir si c’est P1 ou P2 qui passe en premier.